Подробности см. в книге SVEN B. SCHREIBER "Undocumented Windows 2000 Secrets"

x86 (i386) - страничная трансляция

Windows XP 32 bit

Запись каталога страниц – page directory entry (PDE)
Запись таблицы страниц – page table entry (PTE)
Номер страничного блока – page frame number (PFN)

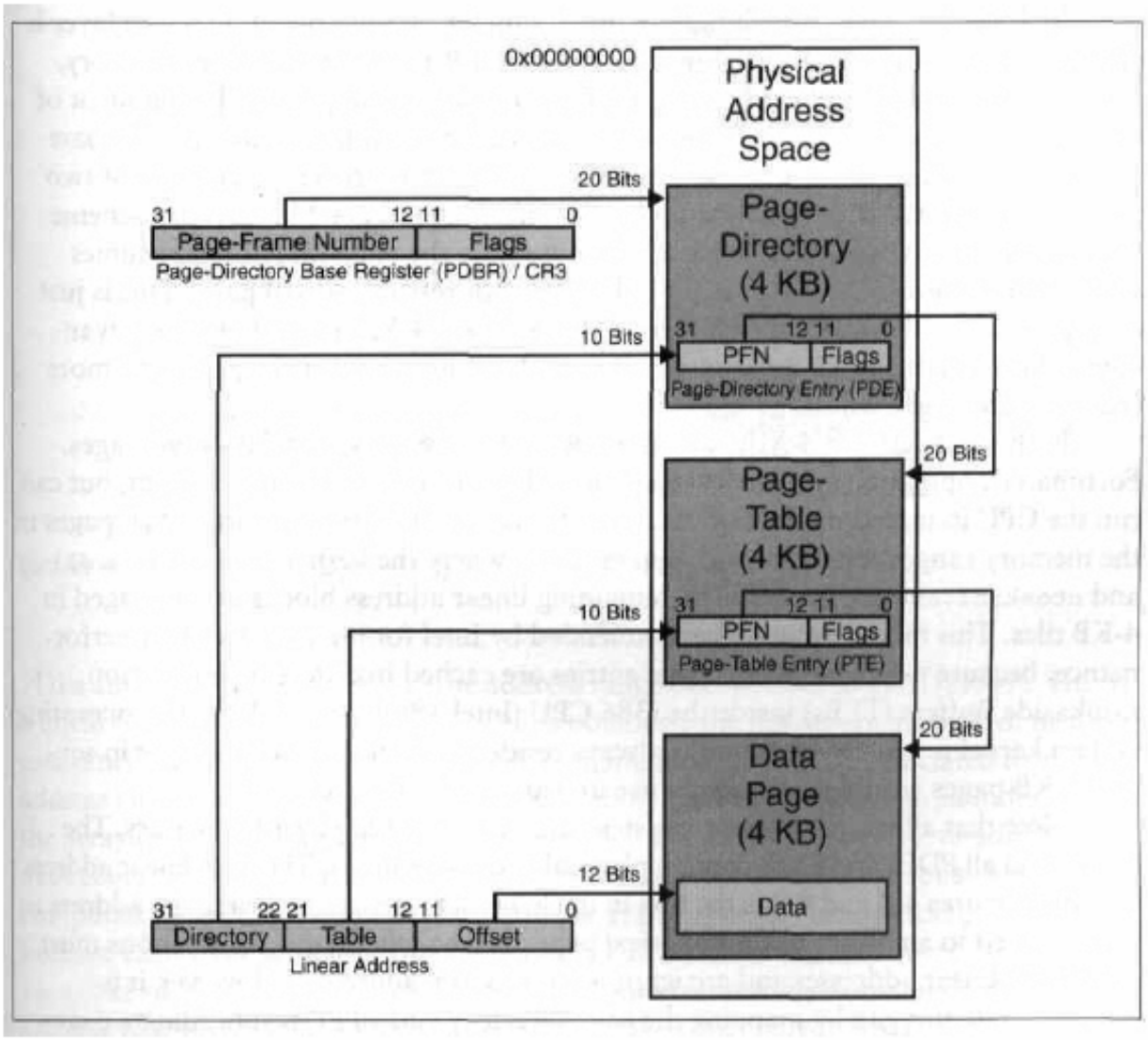

FIGURE 4-3. *Double-Layered Paging with 4-KB Pages*

Используемые структуры из w2k_spy.h

```
// ==================================================================
// INTEL X86 STRUCTURES, PART 2 OF 3
// ==================================================================
```

Copyright © 2003-2011 by int2eh. All rights reserved.

```
typedef struct _X86_PDBR // page-directory base register (cr3)
{
	union
	{
		struct
		{
			DWORD dValue;		// packed value
		};
		struct
		{
			unsigned Reserved1 :  3;
			unsigned PWT	:  1; // page-level write-through
			unsigned PCD	:  1; // page-level cache disabled
			unsigned Reserved2 :  7;
			unsigned PFN	: 20; // page-frame number
		};
	};
}
X86_PDBR, *PX86_PDBR, **PPX86_PDBR;

#define X86_PDBR_ sizeof (X86_PDBR)

// ---------------------------------------------------------------

typedef struct _X86_PDE_4M // page-directory entry (4-MB page)
{
	union
	{
		struct
		{
			DWORD dValue;		// packed value
		};
		struct
		{
			unsigned P	:  1; // present (1 = present)
			unsigned RW	:  1; // read/write
			unsigned US	:  1; // user/supervisor
			unsigned PWT	:  1; // page-level write-through
			unsigned PCD	:  1; // page-level cache disabled
			unsigned A	:  1; // accessed
			unsigned D	:  1; // dirty
			unsigned PS	:  1; // page size (1 = 4-MB page)
			unsigned G	:  1; // global page
			unsigned Available :  3; // available to programmer
			unsigned Reserved  : 10;
			unsigned PFN	: 10; // page-frame number
		};
	};
}
X86_PDE_4M, *PX86_PDE_4M, **PPX86_PDE_4M;

#define X86_PDE_4M_ sizeof (X86_PDE_4M)

// ---------------------------------------------------------------

typedef struct _X86_PDE_4K // page-directory entry (4-KB page)
{
	union
	{
		struct
		{
			DWORD dValue;		// packed value
		};
```

Copyright © 2003-2011 by int2eh. All rights reserved.

```
                struct
                {
                        unsigned P        :  1; // present (1 = present)
                        unsigned RW       :  1; // read/write
                        unsigned US       :  1; // user/supervisor
                        unsigned PWT      :  1; // page-level write-through
                        unsigned PCD      :  1; // page-level cache disabled
                        unsigned A        :  1; // accessed
                        unsigned Reserved :  1; // dirty
                        unsigned PS       :  1; // page size (0 = 4-KB page)
                        unsigned G        :  1; // global page
                        unsigned Available :  3; // available to programmer
                        unsigned PFN      : 20; // page-frame number
                };
        };
}
X86_PDE_4K, *PX86_PDE_4K, **PPX86_PDE_4K;

#define X86_PDE_4K_ sizeof (X86_PDE_4K)

// ---------------------------------------------------------------

typedef struct _X86_PTE_4K // page-table entry (4-KB page)
{
        union
        {
                struct
                {
                        DWORD dValue;           // packed value
                };
                struct
                {
                        unsigned P        :  1; // present (1 = present)
                        unsigned RW       :  1; // read/write
                        unsigned US       :  1; // user/supervisor
                        unsigned PWT      :  1; // page-level write-through
                        unsigned PCD      :  1; // page-level cache disabled
                        unsigned A        :  1; // accessed
                        unsigned D        :  1; // dirty
                        unsigned Reserved :  1;
                        unsigned G        :  1; // global page
                        unsigned Available :  3; // available to programmer
                        unsigned PFN      : 20; // page-frame number
                };
        };
}
X86_PTE_4K, *PX86_PTE_4K, **PPX86_PTE_4K;

#define X86_PTE_4K_ sizeof (X86_PTE_4K)

// ---------------------------------------------------------------

typedef struct _X86_PNPE // page not present entry
{
        union
        {
                struct
                {
                        DWORD dValue;           // packed value
                };
                struct
                {
                        unsigned P        :  1; // present (0 = not present)
```

Copyright © 2003-2011 by int2eh. All rights reserved.

```
                        unsigned Reserved1 :  9;
                        unsigned PageFile  :  1; // page swapped to pagefile
                        unsigned Reserved2 : 21;
                };
        };
}
X86_PNPE, *PX86_PNPE, **PPX86_PNPE;

#define X86_PNPE_ sizeof (X86_PNPE)

// ---------------------------------------------------------------

typedef struct _X86_PE // general page entry
{
        union
        {
                DWORD     dValue; // packed value
                X86_PDBR   pdbr;   // page-directory Base Register
                X86_PDE_4M pde4M;  // page-directory entry (4-MB page)
                X86_PDE_4K pde4K;  // page-directory entry (4-KB page)
                X86_PTE_4K pte4K;  // page-table entry (4-KB page)
                X86_PNPE   pnpe;   // page not present entry
        };
}
X86_PE, *PX86_PE, **PPX86_PE;

#define X86_PE_ sizeof (X86_PE)

// ===================================================================
// INTEL X86 STRUCTURES, PART 3 OF 3
// ===================================================================

typedef struct _X86_LINEAR_4M // linear address (4-MB page)
{
        union
        {
                struct
                {
                        PVOID pAddress;       // packed address
                };
                struct
                {
                        unsigned Offset : 22; // offset into page
                        unsigned PDI    : 10; // page-directory index
                };
        };
}
X86_LINEAR_4M, *PX86_LINEAR_4M, **PPX86_LINEAR_4M;

#define X86_LINEAR_4M_ sizeof (X86_LINEAR_4M)

// ---------------------------------------------------------------

typedef struct _X86_LINEAR_4K // linear address (4-KB page)
{
        union
        {
                struct
                {
                        PVOID pAddress;       // packed address
                };
                struct
                {
```

Copyright © 2003-2011 by int2eh. All rights reserved.

```
                        unsigned Offset : 12; // offset into page
                        unsigned PTI    : 10; // page-table index
                        unsigned PDI    : 10; // page-directory index
                };
        };
}
X86_LINEAR_4K, *PX86_LINEAR_4K, **PPX86_LINEAR_4K;

#define X86_LINEAR_4K_ sizeof (X86_LINEAR_4K)

// ---------------------------------------------------------------

typedef struct _X86_LINEAR // general linear address
{
        union
        {
                PVOID        pAddress; // packed address
                X86_LINEAR_4M linear4M; // linear address (4-MB page)
                X86_LINEAR_4K linear4K; // linear address (4-KB page)
        };
}
X86_LINEAR, *PX86_LINEAR, **PPX86_LINEAR;

#define X86_LINEAR_ sizeof (X86_LINEAR)

// ===================================================================
// INTEL X86 MACROS & CONSTANTS
// ===================================================================
#define X86_PAGES        0xC0000000
#define X86_PTE_ARRAY     ((PX86_PE) X86_PAGES)
#define X86_PDE_ARRAY     (X86_PTE_ARRAY + (X86_PAGES >> PTI_SHIFT))
```

// Итак, мой код!

```
//А вот и моя ф-ция - аналог команды PAGE отладчика Syser:
PVOID Syser_CMD_PAGE(PVOID VirtualAddr)
{
        X86_PDBR regCR3;
        X86_LINEAR_4K LinearAddr;
        PX86_PDE_4K Pde;
        PX86_PTE_4K Pte;
        ULONG Ret = -1;
        PVOID Addr = 0;
        PHYSICAL_ADDRESS PhysicalAddress;

        LinearAddr.pAddress  = (PVOID)VirtualAddr;


        if(!MmIsAddressValid(LinearAddr.pAddress))
                return Ret;

        __asm{
                        push eax
                        mov eax, CR3
                        mov regCR3, eax
                        pop eax
        }

        PhysicalAddress.HighPart = 0;
        PhysicalAddress.LowPart = (regCR3.PFN << 12);

        Addr = MmMapIoSpace(PhysicalAddress, PAGE_SIZE, MmWriteCombined);
```

Copyright © 2003-2011 by int2eh. All rights reserved.

```
        if(!Addr)
                return Ret;

        Pde = Addr;
        Pde = (Pde + LinearAddr.PDI);

        if(!Pde->P){
                MmUnmapIoSpace(Addr, PAGE_SIZE);
                return Ret;
        }

        PhysicalAddress.HighPart = 0;
        PhysicalAddress.LowPart = (Pde->PFN << 12);

        MmUnmapIoSpace(Addr, PAGE_SIZE);

        Addr = MmMapIoSpace(PhysicalAddress, PAGE_SIZE, MmWriteCombined);

        if(!Addr)
                return Ret;

        Pte = Addr;

        Pte = (Pte + LinearAddr.PTI);

        if(Pte->P){
                Ret = (PVOID)((Pte->PFN << 12) + LinearAddr.Offset);
        }

        MmUnmapIoSpace(Addr, PAGE_SIZE);

        return (PVOID)Ret;
}
```

//WBR, Бублик (int2eh@mail.ru)

Copyright © 2003-2011 by int2eh. All rights reserved.